CS3733: Operating Systems

Topics: CPU Scheduling Examples and Simulator (SGG 5.6 and web notes)

Instructor: Dr. Dakai Zhu

Outline
- Reviews on simple CPU Scheduling
- Multi-Level queues
- Multi-level feedback queues
- VAX VMS Scheduling
- Windows NT Scheduling
- UNIX Scheduling Algorithm
- Linux Scheduling Algorithm

Reviews
- Scheduling criteria
  - Waiting time
  - Response time
  - Turnaround time
  - CPU utilization
- Preemptive vs. non-preemptive Scheduling
- Classical CPU Schedulers
  - FCFS
  - RR
  - SJF
  - SRJF

Multi-Level Queues
- Classify jobs into different classes
- Maintain separate queues for each class
- Each queue can have its own scheduling algorithm
- How to choose processes from the queues?
  - From the highest priority non-empty queue
  - Time slice among queues: each queue gets its fixed share
- Examples to assign priorities
  - Foreground vs. background
  - Batch vs. real-time
  - Student vs. administrative

Multi-Level Feedback Queues
- Allow a job to change priority (its queue) dynamically
- Need to specify:
  - Number of queues
  - Scheduling algorithm for each queue
  - Initial queue assignment algorithm
  - An upgrade priority algorithm: enable a process to have higher priority at runtime
  - A downgrade priority algorithm: enable a process to reduce its priority at runtime

VAX/VMS Scheduling
- Use multi-level feedback queue: 32 priority levels
- 16-31 → static real-time queues
  - A process gets its priority when it starts
  - Pre-emptive according to priorities
- 0-15 → dynamic queues
  - Processes in higher number queue are executed first
  - Non-preemptive with quantum
VAX/VMS Scheduling (cont.)

- Each system call has a priority increment
  - When an event occurs a process become executable: its priority = base + increment
- When a process is scheduled to run its priority decreases
  - No lower than the base
- Processes age
  - If spend a long time in a queue increase priority

Windows NT Scheduling (W2000, and XP)

- Similar to VAX
- Interactive processes: wait for keyboard or mouse
  - Get largest priority increment
- Foreground process in the current window get a larger quantum
- Preemptive even on the dynamic queues

UNIX Scheduling Algorithm

- Also use multi-level feedback queues
- A runnable process get a number which queue
- Lower numbers higher priority
  - Negative numbers: system processes cannot be killed by signals
- First process in the lowest nonempty queue run
  - nice to to reduce priority
- Time quantum of 0.1 second (100 milliseconds)
- Priorities are re-calculated once a second
- Non-preemptive except for quantum expiration

UNIX Scheduling Algorithm (cont.)

- Some flavors of Unix interactive process with window focus gets highest priority
- A process’ user-mode priority
- When a process is blocked for an event, it cannot accumulate CPU time
- When a process sleeps for more than 1 second, we re-compute its priority
- Do not preemptive process running in kernel mode
  - Not suitable for real-time systems

Linux Scheduling Algorithm

- Use two separate scheduling algorithms
  - One for time-sharing with focus on fairness
  - One for real-time tasks with absolute priorities
- Time sharing processes: based on a credit system
  - Process has a fixed priority and variable number of credits
  - To choose a process the one with most credits to run
  - Running process loses one credit per timer interrupt, which is removed from CPU when its credits run out
  - If no process has any credit credits = credits/2 + priority

Real-Time Scheduling

- Each process has a priority and a scheduling class
  - Scheduling class: can be FIFO
- The highest priority real-time task runs first
- Real-time tasks have higher priority than time-sharing jobs
  - Time-sharing tasks run only if no runnable real-time tasks
- For tasks with same priority: FCFS
  - For FCFS, a process runs until its I/O operation
  - RR processes do NOT preempt other processes
CPU Scheduling Simulator (web notes)

- Schedulers: FCFS, SJF, PSJF, RR(x)
- Processes
  - Arrival time, all CPU and I/O bursts
- Simulations
  - Number of processes
  - Process: first arrival, inter-arrival time distribution, CPU burst distribution, I/O burst distribution
- Experiment
  - Collection of simulation runs → statistical results or conclusions

Summary

- Reviews on simple CPU Scheduling
- Multi-Level queues
- Multi-level feedback queues
- VAX VMS Scheduling
- Windows NT Scheduling
- UNIX Scheduling Algorithm
- Linux Scheduling Algorithm

Reminding

- Assignment2: due on 02/14